UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/449,021 | 11/24/1999 | HELMUT EMMELMANN | EMME-1000US0 | 5718 |

8685 7590 07/12/2011

DERGOSITS & NOAH LLP
Three Embarcadero Center
Suite 410
SAN FRANCISCO, CA 94111

| EXAMINER | |
|---|---|
| KENDALL, CHUCK O | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 07/12/2011 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 09/449,021
Filing Date: November 24, 1999
Appellant(s): EMMELMANN, HELMUT

Richard A. Nebb Reg. No. 33,540
For Appellant

**EXAMINER'S ANSWER**

This is in response to the Appeal Brief filed 06/09/10 appealing from the Final Rejection mailed 01/21/10 and the Reply Brief filed December 13, 2010 in response to the Examiner's Answer mailed October 13, 2010.

**(1) Real Party in Interest**

A statement identifying the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the Reply Brief is correct.

**(4) Status of Amendments After Final**

In response to an indication of allowability of claims 6, 8, 51-58, 74-96, and 114-128 in the Examiner's Answer mailed October 13, 2010, an amendment after final canceling these claims was December 13, 2010. This amendment after final has been entered.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The statement of the grounds of rejection identified in the Appeals Brief filed June 9, 2010 and the Reply Brief filed are incorrect. The grounds of rejection to be reviewed on appeal are:

(1) Whether claims 1, 2, 22, 23, 41, and 42, rejected under 35 U.S.C. 103(a) as being unpatentable over "Responsive Interaction for a Large Web Application", The Meteor Shower Architecture in the WEBWRITER II Editor written by Arturo Crespo et al. (1996), hereinafter WEBWRITER II, in view of Faustini USPN 5,842,020;. and

(2) Whether claims 26, 30, 32, 33, 59 – 63, 67 – 69 and 71 – 73 rejected under 35 U.S.C. 103(a) as being unpatentable over "Web Writer: A Browsers -Based Editor for Constructing Web Applications" written by Arturo Crespo et al. (1996), Hereinafter "WEBWRITER", in view of Faustini USPN 5,842,020.

**WITHDRAWN REJECTIONS**

The following grounds of rejection are not presented for review on appeal because they have been withdrawn by the examiner:

The rejection of claims 1-6, 8, 22-33, 41-43, 51-96, and 114-128 under 35 U.S.C. 103 (a) as being unpatentable over Web Writer: A Browsers-Based Editor for Constructing Web Applications Crespo et al. Published May 1996 (Hereinafter "Web Writer") in view of Web Writer II, Crespo et al. Published 1997.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the Appeals Brief filed June 9, 2010 is not correct. The copy of the claims in the Reply Brief is correct.

Claims 3-6, 8, 24-25, 27-29, 31, 43, 51-58, 64-66, 70, 74-96 and 114-128 listed in the appendix of the original Appeals Brief were canceled in an amendment filed December 13, 2010. Claims 9-21, 34-40, 44-50 and 97-113 were previously withdrawn as being directed to non-elected subject matter. Claim 7 was previously canceled.

Claims 1-2, 22-23, 26, 30, 32-33, 41-42, 59-63, 67-69, and 71-73 remain pending and are the subject of this appeal.

(**8) Evidence Relied Upon**

| | | |
|---|---|---|
| WEBWRITER I | Crespo et al. | 1996 |
| WEBWRITER II | Crespo et al. | 1997 |

| | | |
|---|---|---|
| FAUSTINI | USPN 5,842,020. | 1997 |

**(9) Grounds of Rejection**

**NEW GROUND(S) OF REJECTION**

(1) claims 1, 2, 22, 23, 41, and 4, are rejected under 35 U.S.C. 103(a) as being unpatentable over "Responsive Interaction for a Large Web Application", The Meteor Shower Architecture in the WEBWRITER II Editor written by Arturo Crespo et al. (1996), hereinafter WEBWRITER II, in view of Faustini USPN 5,842,020;. and

(2) Claims 26, 30, 32, 33, 59 – 63, 67 – 69 and 71 – 73 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Web Writer: A Browsers -Based Editor for Constructing Web Applications" written by Arturo Crespo et al. (1996), hereinafter "WEBWRITER", in view of Faustini USPN 5,842,020.

**NEW GROUND(S) OF REJECTION**

***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

1. Claims 26, 30, 32, 33, 59 – 63, 67 – 69 and 71 – 73, are rejected under 35 U.S.C. 103(a) as being unpatentable over Web Writer: A Browsers -Based Editor for Constructing Web Applications Arturo et al. Published May 1996 (Hereinafter "WEBWRITER") in view of Faustini USPN 5,842,020.

2. As per claim 26, a system having a data network which couples a server computer to a client computer, the server computer running an application to modify documents on the server computer, *(**WEBWRITER, pg. 1 see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor" wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side)* the server computer comprising:

a document store, *(see section pg. 6, "Saving and Loading the Stack" which details saving and loading pages by file names)*

a first software program including instructions for transforming at least one first document retrieved from the document store into a second document having features which permit editing of the first document such that at least a part of the second document appears and functions similar to the run-time view of the first document *(**via calling WEBWRITER which will invoke WebWriter Editor to generate page into two column display having editing features / functions**; **see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2**, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"** wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side)*

a second software program including instructions to receive information from the client computer and instructions to modify the first document stored in the document store

*(**via invoked WebWriter Editor making changes and regenerate page based on edit features**; **see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2**, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"** wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side).*

**WEBWRITER** doesn't explicitly disclose that the modifying of documents is to dynamic documents such that the editing is to running applications for the dynamic documents.

However, Faustini discloses in col. 5:15 - 23, "dynamic editing capability as soon as the component is instantiated or a component editing

request is made...is also able to edit each component's properties and their limits as desired and then observe the edited changes ..."

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine WEBWRITER and Faustini because it would enable editing each component or property that generates dynamic content as desired and then observe the edited changes live within the environment of WEBWRITER.

2. As per claim 30, the system of claim 26, wherein the features include scripts, ***WEBWRITER,*** *Cut / Copy / Insert functionality that inherently function as scripts to insert copy or remove objects from the web page. (see fig 2 of reference)***.**

3. As per claim 32, the system as in claim 26, wherein the features incorporate information regarding the first document into the second document.

***WEBWRITER,*** *wherein insertion points / changes made in the first document are incorporated into the generation of the second document;* ***WEBWRITER see Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the*

*reference: pg. 2, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor" wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side).*

4. As per claim 33, a system as in claim 32, wherein the information incorporated into the second document is used on the client computer in order to send change requests for the first document to the server computer.

***WEBWRITER,*** *wherein insertion points / changes made in the first document are incorporated into the generation of the second document;* ***WEBWRITER see Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor" wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side).*

5. As per claim 59 A software development system have at least one computer running an application for developing web documents, said web documents operating by being transformed into an end user's view upon a request by a web browser, the end user's view being provided to the browser for display on a display device in response to the request, *(**via calling WEBWRITER which will invoke WebWriter Editor to generate page into two column display having editing features / functions**; **see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2, **"Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"** wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side)*

an editor program having instructions for dynamically editing web contents,

*(**via invoked WebWriter Editor making changes and regenerate page based on edit features**; **see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2, **"Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"** wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side)*

a document generator program having instructions for generating generated documents, from web content which look and function similar to the end user's view of the document during editing thereby resulting in the generated document *(**via invoked WebWriter Editor allowing user to make changes and based on executing each user-specified action, Webwriter generates the next page by traversing the content tree and asking all objects to print themselves in a format appropriate for the document area of the editor**; **see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages*

*at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2,* ***"Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor", "...executing each user-specified action, Webwriter generates the next page by traversing the content tree and asking all objects to print themselves in a format appropriate for the document area of the editor..."*** *wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects, based on WebWriter executing the user's action, regenerate / reprint themselves thereby creating a new document on the left side*);

an editor program comprising first instructions for requesting that the document generator program processes a web document during editing thereby resulting in a generated document, and the system comprising instructions for displaying at least some information items contained on said generated document in a view which allows the user to select an item to which a modification function will be applied, the editor program comprising third instructions to modify the web document to perform said modification function (***via invoked WebWriter Editor allowing user to make changes and based on executing each user-specified action, Webwriter generates the next page by traversing the content tree***

***and asking all objects to print themselves in a format appropriate for the document area of the editor; see Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2* ***, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor", "...executing each user-specified action, Webwriter generates the next page by traversing the content tree and asking all objects to print themselves in a format appropriate for the document area of the editor..."*** *wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects, based on WebWriter executing the user's action, regenerate / reprint themselves thereby creating a new document on the left side*).

**WEBWRITER** doesn't explicitly disclose that the modifying of documents is to dynamic documents such that the editing is to running applications for the dynamic documents wherein the dynamic components once edited are dynamically viewed.

However, Faustini discloses in col. 5:15 - 23, "dynamic editing capability as soon as the component is instantiated or a component editing request is made…is also able to edit each component's properties and their limits as desired and then observe the edited changes …"

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine WEBWRITER and Faustini because it would enable editing each component or property that generates dynamic content as desired and then observe the edited changes live within the environment of WEBWRITER.

6. As per claim 60, the software development system of Claim 59 comprising a data network which couples a server computer and a client computer, the document generator program running on the server computer, the editor program at least partly running on the client computer (***WEBWRITER in the Abstract*** *shows the WebWriter Editor program operating in the Browser and a Web generator CGI program on the server, thereby it is obvious that the user interface of the browser which shows the editor allows for part of the editor program to run on the client computer while remaining components run on the server*).

7. As per claim 61, the software development system of claim 60 comprising fourth instructions for execution during the document generation

to collect edit-information (insertion points / handles) for use by the editor program. ***(via calling WEBWRITER which will invoke WebWriter Editor to generate page into two column display having editing features / functions; see Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2,* ***"Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"*** *wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left*

8. As per claim 62, wherein the editor program uses a web browser for displaying said view (***WEBWRITER see Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor" wherein WebWriter Editor left column displays the live document to allow users to edit (insert,*

*remove, etc) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side).*

9. As per claim 63, a software development system of claim 60, comprising instructions for automatically repeating the request that the document generator processes the dynamic web document when required *(**via invoked WebWriter Editor allowing user to make changes and based on executing each user-specified action, Webwriter generates the next page by traversing the content tree and asking all objects to print themselves in a format appropriate for the document area of the editor**; **see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2**, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor", "...executing each user-specified action, Webwriter generates the next page by traversing the content tree and asking all objects to print themselves in a format appropriate for the document area of the editor..."** wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects,

*based on WebWriter executing the user's action, regenerate / reprint themselves thereby creating a new document on the left side*). Faustini discloses in col. 5:15 - 23, "dynamic editing capability as soon as the component is instantiated or a component editing request is made…is also able to edit each component's properties and their limits as desired and then observe the edited changes …"

10. As per claim 67, the software development system of claim 59, wherein said view looks, except for editing features, similar to the end-user view of the generated document *(via WebWriter displays the current page, e.g. expected webpage to be edited, as interpreted HTML together with additional objects, called insertion points and handles, that aid in the editing process)* ***(see pg. 2, "Creating the Template Pages"; pg. 3, "Insertion Points and Handles")***.

11. As per claim 68, the software development system of claim 59 comprising sixth instructions to collect edit information (insertion points / handles) for use by the editor program, said sixth instructions for execution during the document generation ***(via calling WEBWRITER which will invoke WebWriter Editor to generate page into two column display having editing features / functions; see Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages*

*at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2* ***, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"*** *wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left*

12. As per claim 69, the software development system of claim 68, wherein the editor program uses the edit-information to correctly modify the dynamic web document.

***(via calling WEBWRITER which will invoke WebWriter Editor to generate page into two column display having editing features / functions; see Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2* ***, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"*** *wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once*

*changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side.*

13. As per claim 71, wherein the editor program uses a web browser for displaying said view *(**WEBWRITER see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference: pg. 2, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor" wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side).*

**14.** As per claim 72, the software development system of claim 71, wherein the first instructions comprise seventh instructions for initiating a reload in the browser, ***(via calling WEBWRITER which will invoke WebWriter Editor to generate page into two column display having editing features / functions; see Abstract and Introduction** shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections*

*also in the reference* ***: pg. 2, "Creating the Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"*** *wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby* ***creating a new document*** *(hence reload) on the left side).* Faustini discloses in col. 5:15 - 23, "dynamic editing capability as soon as the component is instantiated or a component editing request is made…is also able to edit each component's properties and their limits as desired and then observe the edited changes …"

15. As per claim 73, The software development system of claim 59 wherein the editor program comprises eighth instructions to display information on at least one element of at least one dynamic web document, that is replaced during document generation, without requesting that the document generator program generates a document *(****via invoked WebWriter Editor displaying web page to allow changes to be made and regenerate page based on edit features****; see* ***Abstract and Introduction*** *shows a server-based application, i.e. Web Writer Editor, which runs on the Browser and allows for direct manipulation by a user to displayed web pages at runtime, e.g. modifying dynamic documents), (see the following sections also in the reference* ***: pg. 2, "Creating the***

***Template Pages", pg. 3, "Insertion Points and Handles", pg. 8, "The WebWriter Editor"*** *wherein WebWriter Editor left column displays the live document to allow users to edit (insert, remove, etc) (thereby incorporating editing features in document) and once changes are made, all WebWriter objects regenerate / reprint themselves thereby creating a new document on the left side) It is noted that if no changes are made to a previously regenerated document and the information is solely displayed, it is inherent to the system that no document is regenerated since only changes allow for the document to be regenerated. In addition, all changes automatically initiate a redisplay such that the user does not request anything of the document generator program.").*

***Claim Rejections - 35 USC § 103***

16. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. Claims 1, 2, 22, 23, 41, and 42, rejected under 35 U.S.C. 103(a) as being unpatentable over "Responsive Interaction for a Large Web

Application", The Meteor Shower Architecture in the WEBWRITER II Editor (hereinafter WEBWRITER II) in view of Faustini USPN 5,842,020.

18. As per Claim 1, a computer-readable medium encoded with computer programs editing software applications that run on a data network which couples a server computer and a client computer, wherein the client computer runs a browser program, and whereupon request by the browser program, at least one of the applications generates generated documents for display by the browser program on a display device and responds to the request with the generated documents (***WEBWRITER II abstract "The WebWriter II Editor, a direct manipulation HTML editor that runs in a web browser uses both server-side and client side processing in order to achieve the advantages of both."; pg. 1513, section 4.1.3, "After the user makes an edit to the document, the screen is redisplayed by calling the JavaScript reload() method on the preview frame. This updates the display without requiring any significant interaction with the server..."; pg. 1513, section 4.1.4, "The first step for modifying an object is selecting it; to do this...The JavaScript code also loads into the object properties frame the appropriate HTML file for the class of the selected object. The selection of an object is a small-scale meteor shower...The user can modify those properties and then click the Done button. This triggers***

***a client side update of the document tree, generation of the new document view by reloading the preview frame, and replacement of the contents of the object properties frame with a blank page.")***

a document generator program running at least part of one of the applications and generating the generated documents; said generating documents including additional editing features for interpretation by the browser program ***(via editing of documents in the preview frame, wherein the documents contain components to generate content such components are rerun in order to display the edited document; see the following sections pg. 1513, section 4.1.3,*** *"After the user makes an edit to the document, the screen is redisplayed by calling the JavaScript reload() method on the preview frame. This updates the display without requiring any significant interaction with the server...The first step for modifying an object is selecting it...The selection of an object is a small scale meteor shower. Fig. 7 shows how the process is initiated in the preview frame, requesting the server to send the appropriate HTML file to the object properties frame...The final step is modifying an object… The user can modify those properties and then click the done button. This triggers a client side update of the document tree, generation of the new document view by reloading the preview frame, and replacement of the contents of the object properties frame with a blank page.;* ***pg. 1511, section 4.1.1,*** *"The*

*HTML page sent to the frame could be static HTML...or an HTML page that includes JavaScript code. Pages with JavaScript code can collaborate with one another via global data structures and functions placed in the top level page of the browser..." pg. 1509, section 3, "The preview frame contains the page that is being edited…In editing mode, the WebWriter II Editor displays the current page as interpreted HTML together with additional images called handles…To insert an object, the user selects it in the insertion control frame and fills in its properties."**)**

an editor program dynamically operating on the generated documents displayed by the browser program via the editing features ***(via editing of documents in the preview frame, wherein the documents contain components to generate content such components are rerun in order to display the edited document; see the following sections pg. 1513, section 4.1.3,*** *"After the user makes an edit to the document, the screen is redisplayed by calling the JavaScript reload() method on the preview frame. This updates the display without requiring any significant interaction with the server...The first step for modifying an object is selecting it...The selection of an object is a small scale meteor shower. Fig. 7 shows how the process is initiated in the preview frame, requesting the server to send the appropriate HTML file to the object properties frame...The final step is modifying an object… The user can modify those properties and then click the done*

*button. This triggers a client side update of the document tree, generation of the new document view by reloading the preview frame, and replacement of the contents of the object properties frame with a blank page.;* ***pg. 1511, section 4.1.1,*** *"The HTML page sent to the frame could be static HTML...or an HTML page that includes JavaScript code. Pages with JavaScript code can collaborate with one another via global data structures and functions placed in the top level page of the browser..." pg. 1509, section 3, "The preview frame contains the page that is being edited…In editing mode, the WebWriter II Editor displays the current page as interpreted HTML together with additional images called handles…To insert an object, the user selects it in the insertion control frame and fills in its properties."*.

**WEBWRITER II** doesn't explicitly disclose that the editing is to running applications for generated documents.

However, Faustini discloses in col. 5:15 - 23, "dynamic editing capability as soon as the component is instantiated or a component editing request is made…is also able to edit each component's properties and their limits as desired and then observe the edited changes …" ***The combination of WebWriter II and Faustini allows for the editing of JavaScript***

***code that incorporation with the document tree remaps component data into a new desired HTML representation of handles***

Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine WEBWRITER II and Faustini because it would enable editing each component or property as desired and then observe the edited changes live within the environment of WEBWRITER II.

19. As per claim 2, a computer-readable medium as in claim 1, further encoded with a plurality of components, and wherein the software applications comprise at least one document template capable of containing components, and wherein the editor provides features to insert, modify and delete a component on at least one document template, and wherein the document generator executes selected components on document templates.

**WEBWRITER II,** *section 4.1.1, "The HTML page sent to frame could be static HTML…or an HTML page that includes JavaScript code…The HTML page loaded in the preview frame contains a script that translates the document tree stored at the top window level into an HTML representation with handles (e.g. into a template document for display)* ***via editing of documents in the preview frame, wherein the documents contain components to generate content such components are rerun in order***

***to display the edited document; see the following sections pg. 1513, section 4.1.3,*** *"After the user makes an edit to the document, the screen is redisplayed by calling the JavaScript reload() method on the preview frame. This updates the display without requiring any significant interaction with the server...The first step for modifying an object is selecting it...The selection of an object is a small scale meteor shower. Fig. 7 shows how the process is initiated in the preview frame, requesting the server to send the appropriate HTML file to the object properties frame...The final step is modifying an object… The user can modify those properties and then click the done button. This triggers a client side update of the document tree, generation of the new document view by reloading the preview frame, and replacement of the contents of the object properties frame with a blank page.;* ***pg. 1511, section 4.1.1,*** *"The HTML page sent to the frame could be static HTML...or an HTML page that includes JavaScript code. Pages with JavaScript code can collaborate with one another via global data structures and functions placed in the top level page of the browser..." pg. 1509, section 3, "The preview frame contains the page that is being edited…In editing mode, the WebWriter II Editor displays the current page as interpreted HTML together with additional images called handles…To insert an object, the user selects it in the insertion control frame and fills in its properties.". Faustini discloses in col. 5:15 - 23, "dynamic editing capability*

*as soon as the component is instantiated or a component editing request is made…is also able to edit each component's properties and their limits as desired and then observe the edited changes …" The* ***combination of WebWriter II and Faustini allows for the editing of JavaScript code that incorporation with the document tree remaps component data into a new desired HTML representation of handles****.*

20. As per Claim 22, the claim limitations are substantially similar in content to claim 1 and are therefore rejected for the same rationale.

21. As per claim 23, a computer readable medium as in claim 22, wherein the editor program operates a functional application in an edit mode permitting editing of said application directly in the web browser. **WEBWRITER II,** *section 4.1.1, "The HTML page sent to frame could be static HTML…or an HTML page that includes JavaScript code…The HTML page loaded in the preview frame contains a script that translates the document tree stored at the top window level into an HTML representation with handles (e.g. into a template document for display)* ***via editing of documents in the preview frame, wherein the documents contain components to generate content such components are rerun in order to display the edited document; see the following sections pg. 1513, section 4.1.3,*** *"After the user makes an edit to the document, the screen is redisplayed by*

*calling the JavaScript reload() method on the preview frame. This updates the display without requiring any significant interaction with the server...The first step for modifying an object is selecting it...The selection of an object is a small scale meteor shower. Fig. 7 shows how the process is initiated in the preview frame, requesting the server to send the appropriate HTML file to the object properties frame...The final step is modifying an object… The user can modify those properties and then click the done button. This triggers a client side update of the document tree, generation of the new document view by reloading the preview frame, and replacement of the contents of the object properties frame with a blank page.;* ***pg. 1511, section 4.1.1,*** *"The HTML page sent to the frame could be static HTML...or an HTML page that includes JavaScript code. Pages with JavaScript code can collaborate with one another via global data structures and functions placed in the top level page of the browser..." pg. 1509, section 3, "The preview frame contains the page that is being edited…In editing mode, the WebWriter II Editor displays the current page as interpreted HTML together with additional images called handles…To insert an object, the user selects it in the insertion control frame and fills in its properties.". Faustini discloses in col. 5:15 - 23, "dynamic editing capability as soon as the component is instantiated or a component editing request is made…is also able to edit each component's properties and their limits as desired and then observe*

*the edited changes …" The* ***combination of WebWriter II and Faustini allows for the editing of JavaScript code that incorporation with the document tree remaps component data into a new desired HTML representation of handles****.*

22. As per claim 41, a computer-readable medium as in claim 1, the editor program comprising a client part for execution on the client computer, ***WEBWRITER II abstract "The WebWriter II Editor, a direct manipulation HTML editor that runs in a web browser uses both server-side and client side processing in order to achieve the advantages of both."***

23. As per claim 42, a computer-readable medium of claim 41, wherein the client part comprises instructions for execution during editing that are automatically downloaded from the server computer in a request prior to editing *(****WEBWRITER II****, section 4.1.1, "The user starts the WebWriter II Editor by invoking a CGI script at the server. The server-side CGI script creates an HTML page with three components...The global data includes the document tree which holds the elements of the page that the user is editing...") (In addition* ***Faustini*** *discloses in 5:15 - 23, "dynamic editing capability as soon as the component is instantiated or a component editing request is made…is also able to edit each component's properties and*

*their limits as desired and then observe the edited changes …"; 154: 36-48, "If an editor appears when a component is instantiated then the user instantaneously knows that that particular component is customizable.")*

**(10) Response to Arguments.**

24. The new grounds of rejection continue to rely on WebWriter I and II but in separate grounds of rejection. The combination of WebWriter I and WebWriter II is no longer used in the rejection of any pending claim. In addition, only the editor of both versions of WebWriter is used in the rejections. Neither the Page Generator of WebWriter I or that of WebWriter II is used as the bases of the rejections. Thus, any arguments relating to use of the Page Generator are also moot.

The responses below are to arguments filed in the Appeal's Brief filed June 9, 2010 and Reply Brief December 13, 2010.

**Responses to Arguments Presented in the Appeals Brief filed June 9, 2010**

25. On pages 13- 15, Appellant argues that the Web Writer Editor is only used to create/edit static parts of the document and that placeholders

are used instead of dynamic content during editing. In contrast, Appellant argues that the instant invention is used to edit components which are dynamic parts. The examiner asserts first that it would have been obvious to the person of ordinary skill in the art to display dynamic areas of the document in its actual appearance during editing sessions instead of just the static components in order to provide the user with a realistic version of the document being edited. Faustini as discussed in the body of the rejection provides the means to do so.

26. In section Aii starting on page 15 of Brief, Appellant argues that the WebWriter articles fail to disclose execution of the application during editing. In response, the Examiner first notes that disclosure of the script placeholder during creation of the document does not mean that the script could not be executed during editing, particularly in view of Faustini as applied in the rejection above. In addition, the Examiner directs attention to Figure 2 of the WebWriter article and Figures 5 and 7 of the WebWriter II article disclosing use of a Preview Window is a part of the Editor. In the Preview Window, the current document is displayed as it is being edited. Changes made by the user are shown as updates or redisplays in the view in the Preview. Both articles disclose use of static and dynamic content as part of the document generated using WebWriter tool, see section 1.1 of the

Introduction in WebWriter II and page 1, section "Introduction" of the WebWriter article. Thus, a document created by the WebWriter system could include an application (dynamic content). The examiner takes note that the dynamic content is computed at run time but respectfully asserts that the once the document is created, it would have been obvious to present the entire document with both static and dynamic content in order to show a more realistic view of the document being edited. As the Preview Window provides the page in its current form to the user for editing, it would have been obvious to include dynamic content for editing as well as static content. Faustini provides the means to edit dynamic content.

27. With respect to arguments relating to Placing Dynamic Areas on page 16 of the Brief, the Examiner notes that this section relates creation of document to be presented to the user, see page 2, "The WebWriter application model". As noted on page 3, "Adding and Editing Content", the Editor can also be used to edit an existing document. The Examiner asserts that it would have been obvious to allow the user to use the Editor to modify a page including dynamic content (or an application) in order for the user to incorporate any desired changes and notes that the means to do so exists.

28. In response to arguments relating to views presented by WebWriter during editing and runtime, the Examiner must again respectfully note that the sections cited by Appellant on pages 16-17 relate to creation of a document not to editing of an existing document. The Examiner agrees that during creation of a new document or a new object, the WebWriter system would include a place holder for dynamic content that would be filled in or computed at run time. However, as noted in Figure 2 and the discussion on page 3, "Adding and Editing Content", of the WebWriter article, the user is presented with the document in its current form for editing. It would have been obvious to present any dynamic content currently included in the document and to allow the user to edit that content as well. Faustini provides the means to do so.

29. With respect to arguments made in earlier rejections, the examiner notes that all pending rejections are included in above. Arguments made in response to prior rejections are moot.

30. On pages 18-20 of the Brief, Appellant reiterates arguments relating to editing of an application while running and the WebWriter Page Generator which have been addressed above. Arguments relating the Page Generator are moot as the Page Generator no longer forms the basis of the

rejection. In addition, Appellant argues that the prior art fails to disclose use of editing features in pages generated from at least partly running edited application. The Examiner must respectfully disagree. As discussed above in the body of the rejection, the combination of WebWriter and Faustini clearly discloses editing of a document while it is being displayed. If the document being edited includes dynamic content or applications, it would have been obvious to run this content while the document is being displayed in order to provide a more realistic view to the user. Note also use of handles or editing features in WebWriter. Faustini provides the means to edit the dynamic content. The dynamic content included in the template and computed at run time is used to generate the document that is displayed to the user. Hence it would have been obvious to include editing features in this document for display to the user.

31. Arguments in section B.(i)(e) relate to traversal of rejections or arguments presented in prior actions, the Examiner notes that the rejections relied upon for this appeal are presented above. Arguments relating to previous art rejections are now moot.

32. In section B(ii), Appellant discussed a position taken by the Examiner equating components to buttons and computing content on the fly

to claimed components. The Examiner respectfully notes that this position is not part of the rejection currently forming the basis of the appeal.

33. For these reasons the Examiner asserts that claim 1 is not patentable over the cited prior art.

34. As per claim 22, arguments relating to execution of at least part of an application during editing have been addressed above. As per "the" generated documents, the WebWriter Editor displays the document in its current form for editing. Thus, it generates documents from the template provided originally and operates on these documents.

35. With respect to claims 26 and 32, the basis for the rejection is articulated above. Further, since the document presented in the Preview Window for editing is the current form of the document, it appears and functions similar to the first document (the document before editing).

36. Editing of application or components has been discussed in the rejection above and in comments above.

37. With respect to arguments presented for Claims 59 and 71, editing of a dynamic document (i.e., one with dynamic content or applications) has been addressed above.

38. Arguments relating to claims 5 and 74 are moot as these claims have been canceled.

39. Claims 90 and 92, 114 and 128, as well as 125 and 127 have been canceled. Thus arguments relating to the patentability of this claim are moot.

40. Claims 3-5 have been canceled.

41. With respect to claim 2, arguments relating to the document template are addressed in the rejection wherein changes in the document are related to the document tree to regenerate the document.

42. With respect to claims 41-42, Applicant argues that "[s]pecifically claim 41 read together with the base claim implicitly requires the editor client to work on the documents generated by the document generator program on the server". The examiner respectfully disagrees.

There is nothing in claim 1 that requires that document generator program be run on the server. In fact the only limitation requiring that a particular program be part of a particular location within the network is that the browser program be run on the client computer. Other programs are just required to be part of the network. Neither is such a limitation is found in claims 41-42.

43. Claim 8 has been canceled.

44. With respect to claim 23, editing of an application while the application is being run has been addressed above. As per operation of a functional application in edit mode during editing, the Examiner respectfully asserts since the document being displayed in the Preview Window of the WebWriter-Faustini combination is the current form of the document as interpreted HTML, the displayed document is functional. The obviousness of including dynamic content or applications as part of the document in this window has been addressed above.

45. Claims 24- 25, 27-29, and 43 have been canceled.

46. With respect to claim 30, arguments relating to the editing features including scripts have been addressed in the rejection above wherein capability of cutting, copying and deleting inherently are scripts run on the web page.

47. Claim 31 has been canceled.

48. With respect to claim 33, Appellant presents arguments relating to the incorporation of change requests in documents being applied to other documents. WebWriter teaches the limitation because the user is capable of editing the preview page and via the browser, sending changes remotely such that the new updated webpage is regenerated with the incorporated changes. Therefore, the combination allows for a user to change components that generate dynamic content and for such components to rerun remotely and regenerate a new web page in the preview frame for the user to view.

49. Claim 43 and 53-58 have been canceled.

50. With respect to claim 63, arguments relating to automatically repeating the request that the document generator processes the web

document when required, Webwriter teaches the limitation because the user is capable of editing the preview page via the browser and sending changes remotely such that a new updated webpage is regenerated with the incorporated changes. Therefore, the combination allows for a user to change components, as needed, that generate dynamic content and for such components to be automatically rerun remotely and regenerate a new web page based on the requested changes in the preview frame for the user to view.

51. Claims 64-66 have been canceled.

52. With respect to claim 61, collection of edit information is clearly disclosed in use of handles and the right most portion of the editing screen in the browser to collect editing information input by the user. Claim 68 also requires collection of editing information and additionally requires that the instruction be executed during document generation. Documents are generated during update or reload of documents being editing in the Preview Window based on user input via handles.

53. Arguments relating to Claim 60 are moot as the combination of WebWriter I and II no longer a basis for the rejection.

54. With respect to claim 67, the view in the Preview Window looks similar to the document before editing except that handles for use in editing (i.e., editing features) are included.

55. Claim 70 has been canceled.

56. With respect to claim 73, the limitation was interpreted as one of two ways: First way) the at least one element, e.g. component, was previously edited and regenerated into a new web page and *currently the element is being displayed without making any new changes to such and the display of the element is being made without a user requesting that the document generator program to display the document*; or second) by the element being displayed and edited wherein when any changes are made the user does not request the document to be redisplayed. As detailed above, Webwriter allows for the display and editing of documents. Regarding the first interpretation, the documents are only regenerated when changes occur to the elements. Thus, if the elements are not changed, the web page is already downloaded and is simply displayed and not requested to be redisplayed. Regarding the second interpretation, all changes automatically initiate a redisplay such that the user does not request

anything of the document generator program. Since there was no section in Applicant's specification that details the intended interpretation of this claim, the examiner applied what was known to one of ordinary skill in the art and therefore applied this interpretation in rejecting the claims.

57. Claims 74-128 have been canceled.

58. Arguments relating to claims 5-6, 51-52, 74 are moot as these claims have been canceled.

**Responses to Arguments Presented in the Reply Brief filed December 13, 2010.**

59. Arguments relating to the combination of WebWriter I and WebWriter II as well as the Page Generator are moot in view of new grounds of rejection

60. On page 4 of the Rely Brief, Appellant argues that the WebWriter Editor generates pages during editing but fails to execute the edited applications and must use the WebWriter Page Generator. First, the examiner notes that arguments relating to executing edited applications have been addressed above. Secondly, as the current rejection no longer relies on use of the Page Generator function of either version of WebWriter

nor on the combination of WebWriter I and WebWriter II, arguments relating to this function and to the combination are moot.

61. Re features relating to documents being generated by the document generator which running at least part of the application being edit, this feature has been addressed both in the body of the current rejection and in arguments above.

62. As discussed above, the rejections forming the basis of this appeal, do not rely on use of the WebWriter Page Generator Function. Hence arguments relating to this function are moot including those found in section II page 5 of the Reply Brief.

63. In section IV, Appellant traverses prior arguments made the Examiner relating to running of applications during editing and use of the Page Generator function in WebWriter. The current rejections do not rely on use of the Page Generator Function and disclosure of features relating to running an edited application during editing has been addressed above. The remainder of the Reply goes on to address limitations in specific claims. Most arguments appear to relate to prior rejections and/or to features already addressed above with respect to the current grounds of rejection. Exceptions are addressed below.

64. With respect to claim 30, as explained above in both the rejection and the argument section, the examiner has addressed how the claims are rejected based on the interpretation of the claims.

65. With respect to claim 72, as explained in the rejection, Webwriter teaches editing of web documents and regenerating of the web documents with the incorporated changes. Therefore, the regeneration is a reload of the modified web documents.

66. Regarding Arguments pertaining to the combination of WEBWRITER I and WEBWRITER II, WEBWRITER I has been withdrawn from consideration and therefore arguments directed to that combination are henceforth moot in view of a new grounds of rejection.

67. Regarding Appellants argument in C. iii that WEBWRITER II performs most of the editing operations on the client computer (except for loading the document before editing and saving the document after editing), where as Applicant's editor does the editing in a distributed way. The Examiner disagrees. WEBWRITER II in section 4.1.2 discloses that, operations that have associated JavaScript can be executed locally or ask the server to perform some service. A global variable can modified to change the display mode and then the preview frame (client side display) can be redisplayed to reflect the new mode.

The Global variables are further described by WEBWRITER II in section 4.1.1 last paragraph, to be able to collaborate with the JavaScript and as previously disclosed in section 4.1 the CGI modules (server side) script creates HTML pages with Global Structures which interface with the JavaScript and as such Examiner interprets that to be exhibit being performed in a distributed way such as claimed by Appellant.

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

For the above reasons, it is believed that the rejections should be sustained.

This examiner's answer contains a new ground of rejection set forth in section **(9)** above. Accordingly, appellant must within **TWO MONTHS** from the date of this answer exercise one of the following two options to avoid *sua sponte* **dismissal of the appeal** as to the claims subject to the new ground of rejection:

(1) **Reopen prosecution.** Request that prosecution be reopened before the primary examiner by filing a reply under 37 CFR 1.111 with or without amendment, affidavit or other evidence. Any amendment, affidavit or other evidence must be relevant to the new grounds of rejection. A request that complies with 37 CFR 41.39(b)(1) will be entered and considered. Any request that prosecution be reopened will be treated as a request to withdraw the appeal.

(2) **Maintain appeal.** Request that the appeal be maintained by filing a reply brief as set forth in 37 CFR 41.41. Such a reply brief must address each new ground of rejection as set forth in 37 CFR 41.37(c)(1)(vii) and should be in compliance with the other requirements of 37 CFR 41.37(c). If a reply brief filed pursuant to 37 CFR 41.39(b)(2) is accompanied by any amendment, affidavit or other evidence, it shall be treated as a request that prosecution be reopened before the primary examiner under 37 CFR 41.39(b)(1).

Extensions of time under 37 CFR 1.136(a) are not applicable to the TWO MONTH time period set forth above. See 37 CFR 1.136(b) for extensions of time to reply for patent applications and 37 CFR 1.550(c) for extensions of time to reply for ex parte reexamination proceedings.

Respectfully submitted,

Respectfully submitted,

Chuck Kendall.

June 29, 2011.

/Chuck O Kendall/
Primary Examiner, Art Unit 2192

**A Technology Center Director or designee must personally approve the new ground(s) of rejection set forth in section (9) above by signing below:**

/JACK HARVEY/

Director, Technology Center 2100

Conferees

/Lewis A. Bullock, Jr/
SPE, AU 2193

Gail Hayes,
/Gail Hayes/
SPE 2100